ซอฟต์แวร์ชิพ : WN100H แนวคิดใหม่ของ OOP

มิติใหม่สำหรับผู้เขียนโปรแกรมทั้งหลายที่นิยมเขียนในรูปแบบโอโอพี โดยรวมโปรแกรมงานเป็นหนึ่งเดียวคล้ายตัวไอซี และใช้ชื่อว่าซอฟต์แวร์ชิพ

บุญเลิศ เอี่ยมทัศนา

แนวความคิดเริ่มต้นของการเขียนโปรแกรมในแบบโอโอพีคือ ให้สามารถนำโปรแกรมเก่ามาใช้ได้ แต่ความคิดของมนุษย์ไม่มีวันหยุดนิ่งจึงคิดต่อไปว่า ในโครงงานขนาดใหญ่ ๆ หากแบ่งโปรแกรมออกตามหน้าที่ของมันหรือตามปัญหาแยกกันเขียนแล้วนำมารวมกันในขั้นสุดท้าย จะทำให้ลดระยะเวลาในการเขียนโปรแกรมโดยส่วนรวม

ความคิดต่อไปก็คือ ในอนาคตอาจมีผู้เขียนโปรแกรมในงานใดงานหนึ่งออกจำหน่ายเป็นเบอร์ ๆ เช่นเดียวกับเบอร์ไอซี ผู้ใช้เพียงแต่นำมาประกอบกันเป็นโปรแกรมที่ต้องการ ดังเช่นที่เรานำไอซีของจริงมาประกอบกันเป็นวงจรไฟวิ่ง วงจรเสียงเพลง เป็นต้น และเรียกกันว่า ***ซอฟต์แวร์ไอซี***

แต่คำว่าไอซีทำให้มองเป็น Integrated Software หรือซอฟต์แวร์ที่รวมงานต่าง ๆ ไว้ด้วยกันอยู่ในโปรแกรมเดียวดังที่ดำเนินการกันอยู่แล้วอีกแขนงหนึ่ง สำหรับเรื่องนี้หากเทียบกับทางด้านฮาร์ดแวร์แล้วจะกลายเป็น Discreated Logic คือวงจรแยกส่วนเท่านั้น ผมจึงขอเรียกเป็นซอฟต์แวร์ชิพแทน

ไม่เพียงเพื่อเป็นตัวอย่าง

คำว่างานใดงานหนึ่งดูออกจะกว้างไป ผมขอยกตัวอย่างเรื่องหน้าต่างซึ่งน่าจะเป็นตัวอย่างที่เหมาะสม เพราะเป็นเรื่องที่กล่าวถึงกันมามากพอควร จะได้ไม่ต้องกล่าวถึงตัวมันเองมากนัก ขอเรียนให้ทราบในตอนนี้เลยว่าจะนำโปรแกรมมาลงเพียงบางส่วนเท่านั้น โปรแกรมทั้งหมดผู้อ่านคงต้องพึ่งบริการซอฟต์เมล์ครับ

และเมื่อมองทางด้านซอฟต์แวร์ชิพจะเห็นว่าสามารถสร้างได้หลายเบอร์ เช่น เพื่อใช้กับจอภาพแบบต่าง ๆ ให้ผู้ใช้เลือกซื้อได้ตามต้องการ ในด้านการค้าก็คาดว่าคงจะเป็นที่ต้องการของตลาด เพราะโปรแกรมส่วนมากในปัจจุบันนิยมใช้หน้าต่างกันมาก (คงขายดี) และเพื่อให้ใช้งานได้จริงจัง ผมจึงตั้งชื่อหัวข้อนี้ว่าไม่เพียงเพื่อเป็นตัวอย่าง

ซอฟต์แวร์ชิพเบอร์แรกที่ผมจะนำมาเสนอคือ WN100H หมายถึงซอฟต์แวร์ชิพหน้าต่างหมายเลข 100 ใช้กับการ์ดเฮอร์คิวลิส ที่ให้เป็นหมายเลข 100 ก็ไม่มีความหมายอะไรมากนัก เพียงแต่ต้องการให้เป็นเลข 3 ตัว ตัวแรกหมายถึงคุณลักษณะเฉพาะดังจะกล่าวต่อไป ตัวที่สองหมายถึงครั้งที่ของการปรับปรุงใหญ่ ส่วนตัวที่สามหมายถึงครั้งที่ของการปรับปรุงย่อย

ระดับของผู้เขียนโปรแกรม

ในบทความชุด "ศึกษาภาษาสมอลล์ทอล์ค" ผมแบ่งระดับผู้เขียนโปรแกรมและโปรแกรมออกเป็น 2 ระดับ คือ กิจกรรมกับสื่อความ สำหรับในชุดนี้เพื่อให้เข้ากับบรรยากาศทางด้านฮาร์ดแวร์ขอแบ่งเป็นระดับผู้ผลิตคือผู้ที่เขียนโปรแกรมซอฟต์แวร์ชิพ กับระดับผู้ประกอบคือผู้ที่นำซอฟต์แวร์ชิพไปใช้ในโปรแกรมของตน และอีกระดับหนึ่งคือผู้ใช้ได้แก่ผู้ที่นำโปรแกรมที่ประกอบแล้วไปใช้ ซึ่งไม่จำเป็นต้องรู้ถึงการเขียนโปรแกรมภาษาคอมพิวเตอร์ ทั้งนี้เพื่อการอ้างถึงต่อไปครับ

หน้าต่าง

ในการสร้างชิพต์หน้าต่างผมคิดว่าคำถามแรกคือ เราต้องการทราบว่าทั้งผู้ประกอบและผู้ใช้ต้องการให้หน้าต่างทำอะไรได้บ้าง ซึ่งผมคิดว่าคงจะหนีไม่พ้นเรื่องต่อไปนี้ คือ

- สามารถกำหนดขนาดและตำแหน่งของหน้าต่างได้
- สามารถเลื่อน ขยาย หด หน้าต่างได้
- สามารถกำหนดหน้าต่างได้หลายบาน และเลือกใช้ได้ตามต้องการ
- สามารถแสดงเมนูมีรายการให้เลือกได้ แบบที่เหมาะคือพูลดาวน์เมนู แต่ผมขอใช้คำว่าเมนูเพียงสั้น ๆ เท่านั้น

แต่ความต้องการของแต่ละบุคคลอาจไม่เท่ากัน และสมมติว่าซอฟต์แวร์ชิพนี้ราคาสูง ผมจะสร้างออกมา 3 เบอร์ ได้แก่

WN100H เป็นหน้าต่างขั้นพื้นฐาน ให้ผู้ประกอบสามารถกำหนดหน้าต่างได้ไม่จำกัดจำนวน ในหน้าต่างแต่ละบานผู้ประกอบสามารถกำหนดเมนูได้ตามต้องการ แต่เป็นหน้าต่างที่อยู่กับที่ จะเลื่อน ขยาย หรือหดหน้าต่างไม่ได้

WM100H เช่นเดียวกับ WN100H แต่สามารถเลื่อน ขยาย และหดหน้าต่างได้ จึงได้ทำเมนูเป็น 2 แบบคือ เมนูหน้าต่างเพื่อให้ผู้ใช้สามารถดำเนินการกับหน้าต่างได้ เช่น เลื่อน ขยาย หด ฯลฯ กับเมนูโปรแกรมตามความต้องการของผู้ประกอบ

WE100H เช่นเดียวกับ WM100H แต่ได้เพิ่มการดำเนินการของเอดิเตอร์ (editor) ในส่วนที่เกี่ยวกับจอภาพมีรายละเอียดดังจะกล่าวต่อไป

ยูนิต WN100H

การเขียนโปรแกรมซอฟต์แวร์ชิพตามที่กล่าวมานี้สำหรับเทอร์โบปาสคาล ผมเห็นว่าควรทำเป็นยูนิต และในการจัดจำหน่ายจริงจะนำไฟล์ .TPU ออกจำหน่าย ส่วนไฟล์ .PAS ของยูนิตผู้ผลิตจะสงวนไว้ (ในซอฟต์เมล์ผมให้ไฟล์ .PAS ไป ซึ่งจะต้องคอมพายล์เป็นไฟล์ .TPU ก่อน)

สำหรับยูนิตนี้ผมขอเริ่มด้วยการกำหนดแบบข้อมูลตามรูปที่ 1 เริ่มด้วยการกำหนดแบบข้อมูลของจอภาพเพื่อดำเนินการกับจอภาพโดยตรง (ไม่ผ่านคำสั่ง Write) แบบข้อมูลของเมนู แบบข้อมูลโปรแกรมย่อย (Procedure Type :

สำหรับเรื่องราวของแบบข้อมูลชนิดนี้ ผมได้นำไปเขียนเป็นบทความอีกเรื่องหนึ่ง)

และท้ายสุดคือแบบข้อมูลออบเจ็กต์ เพื่อการเขียนโปรแกรมในแบบโอโอพี (ของเทอร์โบปาสคาลซึ่งคล้ายกับใน C++) แต่ในการเขียนโปรแกรมนี้ผมไม่ได้เขียนในแบบโอโอพีโดยแท้ ดังจะเห็นว่าได้กำหนดส่วนหัวของโปร-

```
UNIT  WN100H;
Interface
uses  crt,prime;

CONST  MenuLen = 127;

TYPE

  OneRowT = ARRAY[1..80] OF RECORD ch : Char; attr : Byte  END;
  ScreenT = ARRAY[1..25] OF OneRowT;
  ScreenPtrT = ^ScreenT;
  RowPtrT = ^OneRowT;

  Str60 = String[60];  Str79 = String[79];
  MenuStr = String[MenuLen];
  MenuArray = ARRAY[1..1] OF MenuStr;
  PtrToMenu = ^MenuArray;
  CharSet = SET OF Char;

  WndWidHiInfo = procedure(wwid,whi : Byte);
  OptionCallT = procedure(var rstr : MenuStr; var ch : Char);

VAR   screen : ScreenT ABSOLUTE $B000:$0000;
      lnebuf : OneRowT;

TYPE
  WindowNorm = OBJECT
    wx1,wx2,wy1,wy2 : Byte;
    xx1,xx2,yy1,yy2 : Byte;
    curx,cury       : Byte;
    difx,dify       : Byte;
    wndwid,wndhi    : Byte;
    InfoWndWidHi    : WndWidHiInfo;
    CallOption      : OptionCallT;
    procedure  Init(px1,px2,py1,py2 : Byte; name : Str60);
    procedure  CalDimension;
    procedure  ConInfoWndWidHiTo(newproc : WndWidHiInfo);
    procedure  ConOptionCallTo(newproc : OptionCallT);
    procedure  ReOpenWindow;
    procedure  ShutWindow;
    procedure  PullDownMenu(px1,py1 : Byte; menu : PtrToMenu;
                                    var retstr : MenuStr);
    procedure  Prompter(prompt : MenuStr; var retstr : MenuStr);
    procedure  GetOptByNum(optstr : MenuStr; var num : Byte);
               END;


PROCEDURE  DummyOptionCall(var rstr : MenuStr; var ch : Char);
PROCEDURE  DrawFrame(xx1,xx2,yy1,yy2 : Byte;name : Str60);
PROCEDURE  CalWindowSize(xx1,xx2,yy1,yy2 : Byte; var wndsize : Word);
PROCEDURE  GetPict(x1,x2,y1,y2 : Byte; buf : RowPtrT);
PROCEDURE  PutPict(scrn : ScreenPtrT; x1,x2,y1,y2 : Byte; buf : RowPtrT);
PROCEDURE  PaintRow(x1,x2,y,att : Byte);
PROCEDURE  OutStrTo(scrn : ScreenPtrT; lne : Byte; astr : String;att : Byte);
PROCEDURE  CursorOn;
PROCEDURE  CursorOff;
```

รูปที่ 1 แสดงส่วนอินเตอร์เฟสของยูนิต WN100H

แกรมย่อยจำนวนหนึ่งไว้ในส่วนอินเตอร์เฟสด้วย เพื่อให้โปรแกรมที่ใช้งานยูนิตนี้สามารถเรียกใช้ได้ หากต้องการให้เป็นแบบโอโอพีแท้ ๆ จะต้องนำไปไว้ในออบเจ็กต์ด้วย แต่ผมคิดว่ามันเกะกะ

จากแบบข้อมูลดังกล่าวสามารถเขียนโปรแกรมได้ดังนี้คือ

ในสกุล WindowNorm

สิ่งที่เทอร์โบปาสคาลทำให้สับสนเล็กน้อยคือ เทอร์โบปาสคาลใช้คำว่า Object ในการกำหนดแบบข้อมูล แต่ความหมายของ Object ในโอโอพีจะมีความหมายเป็นของตนเอง ที่จริงแล้วเทอร์โบปาสคาลควรใช้คำว่า Class หรือคำอื่นใดก็ได้ที่ไม่ใช่ Object เมื่อเป็นเช่นนี้ในบทความของผมต่อไปนี้จะใช้คำว่าออบเจ็กต์ตามนัยของโอโอพี นอกจากจำเป็นจึงจะใช้คำว่าแบบข้อมูลออบเจ็กต์ตามนัยของเทอร์โบปาสคาล

เมื่อเรากำหนดแบบข้อมูลออบเจ็กต์แล้ว ก็จะเป็นการกำหนดสกุล (Class) ตามนัยของโอโอพี (เช่นข้อความว่า "ในสกุล WindowNorm" กับ "ในแบบข้อมูลออบเจ็กต์ WindowNorm" มีความหมายตรงกัน) ซึ่งภายในสกุลนี้มีเรื่องราวที่จะต้องดำเนินการคือ

● ตัวแปร

ในการเขียนโปรแกรมแบบโอโอพีนั้นจะถือออบเจ็กต์เป็นหลัก ในกรณีนี้ออบเจ็กต์ของเราคือหน้าต่าง ดังนั้นเราจะต้องเตรียมตัวแปรไว้สำหรับเก็บข้อมูลของหน้าต่าง ข้อมูลของหน้าต่าง (และตัวแปรที่กำหนดให้) คือ จุดโคออร์ดิเนตของมุมบนซ้าย (wx1, wy1) จุดโคออร์ดิเนตของมุมล่างขวา (wx2, wy2) และตำบลที่อยู่ของเคอร์เซอร์ (curx, cury)

เมื่อเป็นเช่นนี้ออบเจ็กต์ของเราคือหน้าต่าง ซึ่งเป็นได้หลายขนาดและหน้าต่างหนึ่ง ๆ ก็คือออบเจ็กต์หนึ่ง ๆ นั่นเอง แต่เรามองเป็นภาพรวม เช่น รถยนต์จะหมายถึงรถยนต์โดยทั่ว ๆ ไปไม่จำกัดขนาด ชนิด รุ่น ฯลฯ

ส่วนตัวแปรอื่น ๆ กำหนดไว้เพื่อความสะดวกในการเขียนโปรแกรมจะได้ไม่ต้องบวก ลบ คูณ หารกันตลอดเวลา และตัวแปรโปรแกรมย่อยคือ

- (xx1,yy1) กับ (xx2,yy2) คือ จุดโคออร์ดิเนตของกรอบหน้าต่าง
- difx กับ dify คือผลต่างของจุดโคออร์ดิเนตในแนวแกน x และแกน y ของหน้าต่างตามลำดับ
- wndwid กับ wndhi คือความกว้างและความสูงของหน้าต่างตามลำดับ
- InfoWndWidHi เพื่อการเชื่อมต่อในแบบไอซี แจ้งความสูงและความกว้างของหน้าต่าง ส่วนของรายละเอียดจะได้กล่าวต่อไป
- CallOption อยู่ในเรื่องของเมนูมีรายละเอียดจะกล่าวต่อไปเช่นกัน

● กิจกรรม

เมื่อเรากำหนดออบเจ็กต์แล้วจะต้องกำหนดกิจกรรม (โปรแกรมย่อย) ที่จะดำเนินการกับออบเจ็กต์นั้น ซึ่งกิจกรรมที่จะดำเนินการกับออบเจ็กต์หน้าต่างโดยตรงได้แก่

- Init เตรียมค่าต่าง ๆ และเปิดใช้หน้าต่าง
- Shut Window ปิดหน้าต่าง (จากบานนี้ไปใช้บานอื่น)
- ReOpenWindow เปิดใช้หน้าต่างใหม่

สำหรับกิจกรรมที่เหลือเป็นกิจกรรมเสริมในการใช้งานหน้าต่าง ได้แก่

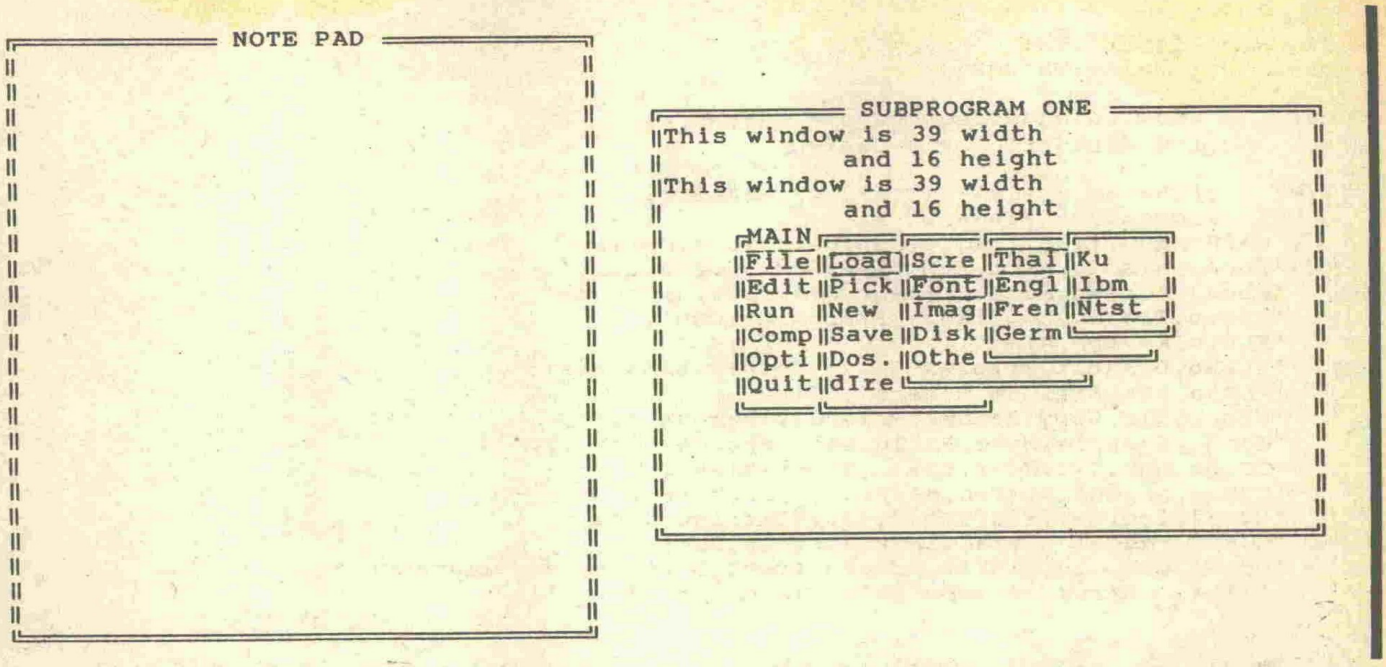
- PullDownMenu แสดงเมนูในแบบพูลดาวน์ให้เลือก ได้ผลเป็นตัวอักษรใหญ่ของรายการที่ให้เลือก
- GetOptByNum แสดงเมนูในแบบพูลดาวน์เช่นกัน แต่ได้ผลเป็นตัวเลขลำดับที่ของรายการที่ให้เลือก
- Prompter เพื่อแสดงข้อความเตือนผู้ใช้ และหากต้องการอาจเป็นคำถามที่ต้องการคำตอบเป็นสตริง เช่น ชื่อไฟล์
- CalDimension เตรียมไว้สำหรับหน้าต่างรุ่นต่อไปจะคำนวณค่า wndwid และ wndhi ใหม่ แต่จุดประสงค์หลักคือเพื่อส่งค่าทั้งสองนี้ไปทาง InfoWndWidHi
- ConInfoWndWidHiTo เชื่อม InfoWndWidHi เข้ากับโปรแกรมของผู้ประกอบ เพื่อให้ทราบค่าความกว้างและความสูงของหน้าต่าง
- ConOptionCallTo เชื่อม CallOption เข้ากับโปรแกรมของผู้ประกอบ รายละเอียดอธิบายไว้ในเรื่องพูลดาวน์เมนู

โปรแกรมย่อยของยูนิต

ดังที่กล่าวแล้วว่าผมไม่ได้เขียนโปรแกรมนี้ในแบบโอโอพีโดยแท้ จึงได้แยกโปรแกรมส่วนหนึ่งออกมาไว้นอกสกุลดังที่เห็นในรูปที่ 1 ซึ่งจะกล่าวในรายละเอียดต่อไป

การทำงานของโปรแกรม

เรื่องราวภายในหัวข้อนี้ผมเขียนขึ้นสำหรับท่านที่ต้องการศึกษาทำความ



รูปที่ 2 ผลของการรันโปรแกรมตัวอย่าง แสดงหน้าต่าง 2 บาน บานทางขวามือที่เห็นเป็นกรอบเล็ก ๆ คือเมนูที่ซ้อนทับกันอยู่

เข้าใจโปรแกรมย่อยแต่ละโปรแกรมในดิสค์ สำหรับท่านที่ต้องการใช้งานยูนิตนี้ก็มีแต่เรื่องเมนูต่าง ๆ เท่านั้นครับ เพราะผมได้แทรกการใช้งานไว้ด้วย โปรแกรมที่ผมขอนำมากล่าวถึงคือ

Init

เตรียมค่าต่าง ๆ ให้แก่ตัวแปรดังกล่าวด้วยค่าที่ผู้เขียนโปรแกรมประกอบผ่านค่ามาให้ ตีกรอบพร้อมทั้งให้ชื่อหน้าต่าง (name) ด้วย DrawFrame แล้วเปิดใช้หน้าต่างด้วย ReOpenWindow ซึ่งจะทำให้แสดงภาพกรอบหน้าต่างด้านบนเป็นตัวดำบนพื้นขาว

ShutWindow

เมื่อต้องการยุติการใช้หน้าต่างนี้ชั่วคราว เช่น ไปเปิดใช้หน้าต่างอื่น ให้ปิดหน้าต่างด้วย ShutWindow ซึ่งจะทำให้ชื่อหน้าต่างเป็นตัวขาวบนพื้นดำ และเปิดจอภาพทั้งจอเป็นหน้าต่างด้วย Window

PullDownMenu

ขอให้ดูหน้าตาของเมนูสมมติในรูปที่ 2 ก่อนครับ ช่องซ้ายสุดคือเมนูหลัก มีเมนูรองและรองลงไปตามที่ผู้ใช้เลือกเรียงไปทางขวา ในการสร้างเมนูนี้ผู้เขียนโปรแกรมประกอบจะต้องเตรียมเมนูและรายการต่าง ๆ โดยทำเป็นอะเรย์ของสตริงดังแสดงในรูปที่ 3 ซึ่งผมได้กำหนดไว้ว่า

● บรรทัดแรกคือสตริงแรกเป็นเมนูหลัก หากไม่มีเมนูรองคงมีบรรทัดเดียว

● ในแต่ละบรรทัดประกอบด้วยรายการของเมนูนั้น ซึ่งแยกจากกันด้วยเครื่องหมาย , นั่นคือ ในแต่ละรายการไม่จำเป็นต้องเขียนติดกันอาจเว้นวรรคได้ และจะมีจำนวนรายการเท่าใดก็ได้ แต่จะถูกจำกัดด้วยความยาวของสตริง คือ 127 อักขระ (ซึ่งผมเห็นว่าหากให้ยาวกว่านี้ก็จะเปลืองเนื้อที่ในหน่วยความจำมากเกินไป)

● รายการใดที่มีเมนูรองให้ต่อท้ายด้วยจุดจุด เช่น File.. แล้วปิดท้ายด้วย , ทันที (ไม่ให้วรรค)

● การเลือกรายการจะให้ผู้ใช้กดคีย์ลูกศรเลื่อนเคอร์เซอร์ทีบไปยังรายการที่ต้องการ แล้วกดคีย์ Enter หรือกดคีย์อักษรที่แสดงเป็นอักษรใหญ่ตัวแรกของรายการนั้น เช่น sAns serif ให้กดคีย์ A

● ในแต่ละบรรทัด (สตริง) รายการแรกมีความหมายพิเศษคือ สำหรับเมนูหลักจะใช้เป็นชื่อเมนู เช่น คำว่า MAIN MENU (ในรูปที่ 3) หากไม่ต้องการให้ขึ้นต้นสตริงด้วย ,

● สำหรับรายการแรกของบรรทัด (สตริง) อื่น ๆ จะหมายถึงรายการในเมนูแม่ของเมนูนั้น เช่น ในบรรทัดที่ 2 คือ Foooo หมายถึงรายการ File ในเมนูหลัก (อักษร o ตัวเล็กผมใส่ไว้เพื่อให้ดูง่ายเท่านั้นครับ) บรรทัดที่ 3 คือ FLooo หมายถึงรายการ File และ Load ซึ่ง

```
PROGRAM  TestW100H;
uses  crt,prime,WN100H;

var    window1,window2,actwnd : WindowNorm;
       opt : MenuStr;  ch : Char;

TYPE   PrgMenuT = ARRAY[1..15] OF MenuStr;
CONST  PrgMenu1 : PrgMenuT =
  ('MAIN MENU,File..,Edit,Run,Compile..,Options..,Quit',
   'Foooo,Load..,Pick,New,Save,Dos..,dIrectory',
   'FLooo,Screen,Font..,Image,Diskfile..,Others',
   'FLFoo,Thai..,English..,French,German',
   'FLFTo,Ku,Ibm,Ntst',
   'FLFEo,Default,Triplex,Small,Gothic,sAns serif',
   'FLDoo,Pas,Exe,Com,Others',
   'FDooo,Dir,Copy,Rename,Make dir,Remove dir',
   'Coooo,Compile,Make,Build,Get info,Options..',
   'COooo,One,Two,Four,Six..,Eight,Nine',
   'COSoo,sixOne,sixTwo,sixFour..,sixNine',
   'COSFo,fourOne,fourTwo..,fourFour,fourNine',
   'COSFT,Pascal,C,Smalltalk,Fortran,Basic',
   'Ooooo,Compile,Linker,Environtment,Directories,Parameters',
   '*****,  Error in menu data can not find....');
```

รูปที่ 3 เป็นส่วนหนึ่งของโปรแกรมตัวอย่าง ในที่นี้ต้องการแสดงการเตรียมข้อมูล PrgMenu1 เพื่อใช้ในการแสดงเมนู ดังรูปที่ 2 แสดงว่าผู้ใช้ได้เลือกรายการ File, Load, Font และ Thai ตามลำดับ

- ผู้เขียนโปรแกรมประกอบต้องดำเนินการตามนี้ และอาจจะถือว่าเป็นชื่อเมนูก็ได้ เพราะในโปรแกรมจะใช้ในการค้นหาสตริง คือ เมื่อผู้ใช้เลือกรายการใด ผมจะเก็บอักษรใหญ่ของรายการนั้นไว้ในสตริง retstr และนำไปเทียบหาเมนูของรายการนั้น
- เช่นเลือก File จะได้ F ไปเทียบกับเมนู Foooo หากต่อไปผู้ใช้เลือก Load จะได้ FL นำไปเทียบหาเมนู FLooo และต่อ ๆ ไป เช่น ตามรูปที่ 2 ค่าใน retstr จะเป็น FLFT นำมาเทียบหาเมนู FLFTo นำมาแสดงเป็นรายการ KU, IBM และ NTST

● บรรทัดสุดท้ายที่ให้ค่าไว้เป็น ***** นั้นก็เพื่อหากมีข้อผิดพลาดในชื่อเมนูดังกล่าวมา จะค้นมาหยุดที่บรรทัดนี้แสดงข้อผิดพลาด ซึ่งควรตรวจพบในขั้นตอนของการทดสอบโปรแกรม ผู้ใช้ไม่ควรมีโอกาสเห็นข้อความนี้

● หากผู้ใช้ไม่ต้องการเลือกรายการใดในเมนูนั้นให้กดคีย์ Esc จะลบเมนูนั้นกลับไปที่เมนูก่อนหน้า หากกด Esc ที่เมนูหลักจะกลับไปยังโปรแกรมที่เรียกมา โดยได้ค่าใน retstr เป็นสตริงว่าง

● หากผู้ใช้เลือกรายการที่ไม่มีเมนูรองจะเก็บตัวอักษรใหญ่ของรายการนั้นใน retstr แล้วลบภาพเมนูทั้งหมดกลับไปยังโปรแกรมที่เรียกมา เช่น ตามรูปที่ 2 หากผู้ใช้เลือกรายการ Ku ค่าที่ได้กลับไปใน retstr จะเป็น FLFTK ซึ่งโปรแกรมที่เรียกมาจะต้องนำไปดำเนินการต่อไป

● แต่ผมเห็นว่าในบางงานผู้เขียนโปรแกรมประกอบอาจต้องการให้คงอยู่ในเมนูนั้นก่อน เช่น ในเทอร์โบปาสคาล เมื่อเลือกรายการ Change directory ในเมนู File และได้ป้อนไดเรคทอรีใหม่แล้ว เมื่อกลับไปจะคงอยู่ในเมนู File (ไม่ได้กลับไปยังเอดิเตอร์) จึงได้เตรียมการในเรื่องนี้ไว้โดย

- ขอให้ย้อนกลับไปดูรูปที่ 1 ได้กำหนดแบบข้อมูล OptionCallT ไว้เป็นข้อมูลแบบโปรแกรมย่อย
- นำมากำหนดเป็นแบบข้อมูลให้แก่ตัวแปร CallOption ใน WindowNorm
- ในยูนิตนี้ได้เขียนโพรซีเดอร์ DummyOptionCall ไว้เป็นโปรแกรมว่าง คือมีแต่ begin กับ end;
- ใน Init ได้ให้ค่าโพรซีเดอร์นี้แก่ CallOption เผื่อไว้ หากผู้เขียนโปรแกรมประกอบไม่ใช้สิ่งอำนวยความสะดวกนี้จะได้ไม่มีการดำเนินการใด
- ในโปรแกรม PullDownMenu เมื่อผู้ใช้เลือกรายการใดจะนำมาตรวจดู หากเป็นรายการที่ไม่มีเมนูรอง (ของเดิมจะให้กลับไป

ทันที) จะเรียกใช้ CallOption ซึ่งจะผ่านไปยังโปรแกรมของผู้เขียนโปรแกรมประกอบ หากได้ดำเนินการดังจะกล่าวต่อไปในหัวข้อตัวอย่างโปรแกรมใช้งาน

เรื่องราวของ PullDownMenu ตามที่กล่าวมานี้ ในโปรแกรมจริงผมให้เรียกใช้ PullMenu ดำเนินการให้ เพื่อความสะดวกในการปรับปรุงครั้งต่อไป

สำหรับทางด้านการใช้งานขอให้ดูที่ส่วนหัวของโปรแกรมนี้ (รูปที่ 1) ได้กำหนดตัวแปร Menu เป็นตัวแปรพอยน์เตอร์ไปยังอะเรย์ของสตริง เพื่อรับข้อมูลของเมนู แต่ตามรูปที่ 2 ได้กำหนดข้อมูลของเมนูไว้เป็นอะเรย์ของสตริง ดังนั้นค่าที่จะผ่านไปให้คือแอดเดรสของอะเรย์นี้ ดังจะเห็นในตัวอย่างโปรแกรมใช้งาน

ในทำนองเดียวกันหากไม่มีเมนูรองคือ มีเพียงสตริงเดียวก็ให้ผ่านแอดเดรสของสตริงนั้นไปให้ วิธีที่ปลอดภัยคือกำหนดตัวแปรสตริงไว้เป็นโกลบอล ให้ค่าตัวแปรนี้ แล้วผ่านแอดเดรสของตัวแปรไปให้

สำหรับค่า px1 กับ py1 คือค่าโคออร์ดิเนตของมุมบนซ้ายของเมนู เป็นค่าในหน้าต่างคือ (1,1) ที่มุมบนซ้ายของหน้าต่าง

Prompter

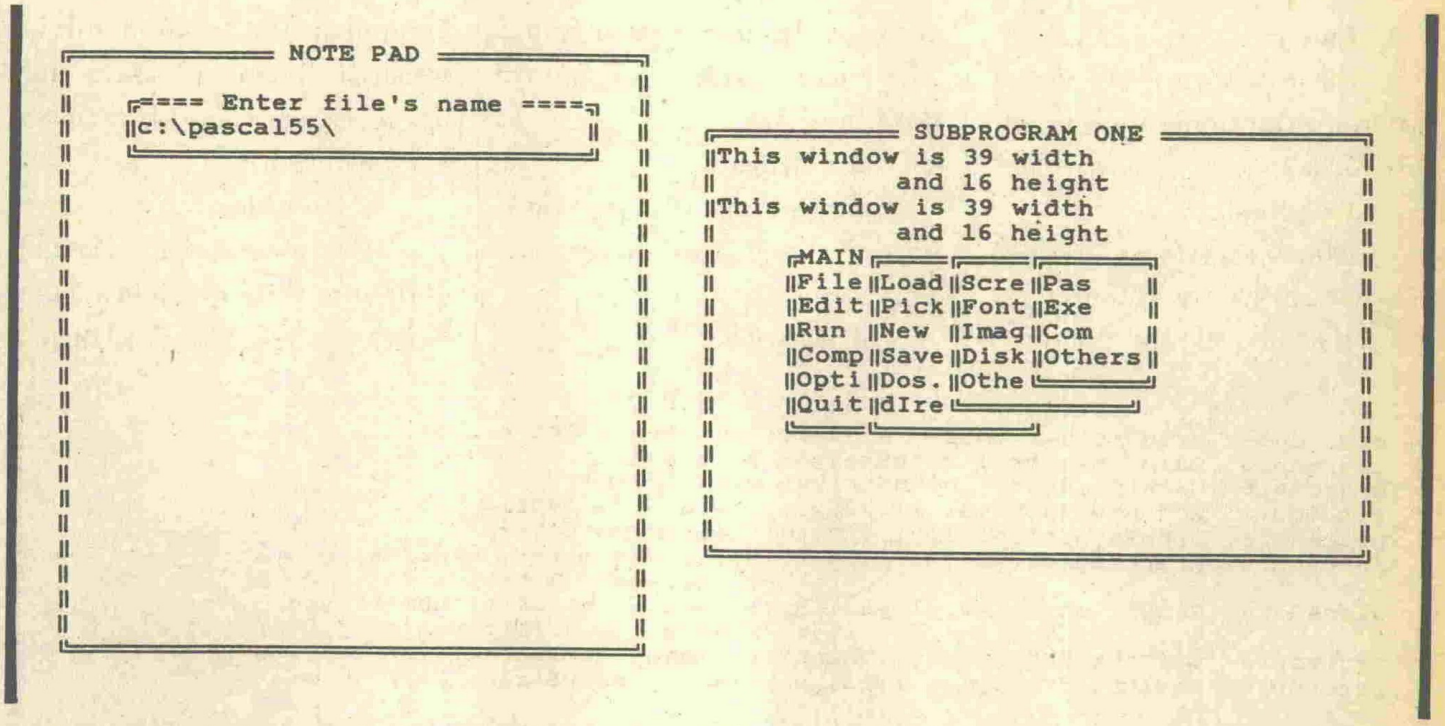
นอกจากเมนูแล้วในการใช้งานหน้าต่างผู้เขียนโปรแกรมประกอบอาจต้องการตั้งคำถามให้ผู้ใช้ตอบเป็นข้อความ จึงได้เตรียมกิจกรรมนี้ไว้ให้ ซึ่งความจริงก็คือเมนูแบบ 2 บรรทัดนั่นเอง แต่อ่านสตริงแทนการเลือกรายการ

ในส่วนหัวของโปรแกรมนี้ prompt คือข้อความที่จะแจ้งให้ผู้ใช้ทราบ ดังรูปที่ 4 คือข้อความว่า Enter file's name หากไม่มีให้เป็นสตริงว่าง สำหรับ retstr เพื่ออ่านสตริงที่ผู้ใช้ป้อนกลับมา แต่ผมได้อ่านสตริงที่ผู้ใช้ป้อนกลับมา แต่ผมได้กำหนดให้ส่งข้อความตัวอย่างไปแสดงได้ด้วยตามรูปคือ

C: \ PASCAL55 \

และกำหนดให้ดำเนินการดังนี้คือ

- หากผู้ใช้กดคีย์ Esc แสดงว่าไม่ต้องการป้อนข้อความใด จะได้ retstr เป็นสตริงว่างกลับไป
- หากผู้ใช้กดคีย์ Enter แสดงว่าต้องการข้อความเดิม จะได้ข้อความเดิมใน retstr กลับไป
- หากผู้ใช้กดคีย์ End แสดงว่าต้องการป้อนข้อความต่อท้าย ให้เลื่อนเคอร์เซอร์ไปท้ายข้อความเดิมใน retstr แล้วอ่านคีย์มาเก็บต่อท้ายใน retstr จนกว่าผู้ใช้จะกดคีย์ Enter
- หากผู้ใช้กดคีย์อื่นใดนอกเหนือ



รูปที่ 4 ผลการรันโปรแกรมตัวอย่าง ในหน้าต่างทางซ้ายคือการทำงานของโพรซีเดอร์ Prompter

จากนี้แสดงว่า ผู้ใช้ต้องการป้อนข้อความใหม่ทั้งหมด ให้ยกเลิกข้อมูลใน retstr เดิม แล้วอ่านคีย์มาเก็บใน retstr จนกว่าผู้ใช้จะกดคีย์ Enter

DrawFrame

เพื่อตีกรอบหน้าต่าง เมนูและ Prompter โดยมีจุดโคออร์ดิเนตของกรอบเป็น (xx1,yy1) และ (xx2,yy2) และแสดงชื่อใน name ที่กรอบด้านบน หากไม่ต้องการ ให้ name เป็นสตริงว่าง

CalWindowSize

เพื่อคำนวณจำนวนไบต์ของหน่วยความจำที่จะเก็บภาพบนจอภาพในกรอบ (xx1,yy1) กับ (xx2,yy2) รวมทั้งกรอบ ให้ค่าที่คำนวณได้แก่ wndsize

GetPict

เก็บภาพบนจอภาพภายในกรอบที่กำหนดด้วย (x1,y1) กับ (x2,y2) รวมทั้งกรอบไว้ในหน่วยความจำฮีพตามที่ buf ชี้อยู่

PutPict

นำภาพในฮีพตามที่ buf ชี้อยู่มาลงบนจอภาพในกรอบที่กำหนดด้วย (x1, y1) กับ (x2,y2)

PaintRow

เพื่อเปลี่ยนค่าแอตทริบิวต์ของข้อมูลจอภาพที่แถว y จากคอลัมน์ x1 ถึง x2 ด้วยค่า att เช่น เปลี่ยนการแสดงอักษรจากขาวเป็นดำบนพื้นขาว หรือให้กะพริบ เป็นต้น

OutStrTo

เพื่อแสดงข้อความในสตริง astr ทางจอภาพโดยตรงที่แถว lne ด้วยค่าแอตทริบิวต์ att

CursorOff และ CursorOn

เพื่องดการแสดงภาพเคอร์เซอร์กะพริบและนำมาแสดงใหม่

โปรแกรมย่อยในยูนิต

เป็นโปรแกรมย่อยทั่วไปที่ใช้ในการสนับสนุนการทำงานของโปรแกรมกิจกรรมคือ โปรแกรมในออบเจ็กต์ WindowNorm ดังกล่าว โดยแสดงเฉพาะส่วนหัวไว้ในรูปที่ 5 มีโปรแกรมย่อยดังนี้

- **GetName**

ตามที่กล่าวแล้วว่ารายการแรกของแต่ละเมนูคือชื่อเมนู PullDown Menu จะผ่านสตริงที่เป็นเมนูมาให้ GetName ใน mstr เพื่อแยกชื่อเมนูให้แก่ nme และตัดชื่อเมนูออกจากสตริง mstr ด้วย

- **CalHigh**

เพื่อหาความสูง (จำนวนบรรทัด) ของเมนูใน mstr ให้แก่ hi ด้วยการทับเครื่องหมาย ,

- **CalWide**

เพื่อหาความกว้างของเมนูใน mstr ให้แก่ wid คือจำนวนอักขระของรายการที่ยาวที่สุด รวมทั้งชื่อเมนูด้วย

- **AdjustWindow**

จากจุดโคออร์ดิเนต (x1,y1) ตามที่ผู้เขียนโปรแกรมประกอบกำหนด รวมทั้งค่าความกว้าง (wid) และความสูง (hi) โปรแกรมย่อยนี้จะนำมาคำนวณเป็นกรอบหน้าต่างของเมนู และปรับแต่งไม่ให้ตกจอภาพ (คืออาจอยู่นอกหน้าต่างของตนได้)

- **DisplayOption**

นำรายการใน mstr มาแสดงบนจอภาพบรรทัดละ 1 รายการ โดยให้ต้นบรรทัดแรกอยู่ที่จุด (x1,y1)

- **PrepMenu**

เพื่อการแสดงเมนูคือ เก็บภาพบนจอภาพเดิมที่เมนูจะมาทับไว้กับตัวแปรพอยน์เตอร์ buf ของ MenuDat (เรื่องของตัวแปร MenuDat จะนำไปกล่าวในหัวข้อ "ตัวแปรในยูนิต") และเก็บข้อมูลของเมนูอื่น ๆ ไว้กับตัวแปรใน MenuDat ตีกรอบของ Menu และนำรายการมาแสดงด้วยโปรแกรมย่อยต่าง ๆ ดังกล่าวมาแล้ว

- **GetOption**

หลังจากที่แสดงรายการแล้วจะให้ผู้ใช้เลือกรายการด้วยคีย์ลูกศรแล้วกด Enter หรือกดคีย์ที่เป็นอักษรใหญ่ตัว

```
procedure  GetName(var mstr : MenuStr; var nme : Str60);
procedure  CalHigh(mstr : MenuStr;var hi : Byte);
procedure  CalWide(mstr : MenuStr;var wid : Byte);
procedure  AdjustWindow(var x1,x2,y1,y2,wid,hi : Byte);
procedure  DisplayOption(x1,y1 : Byte; mstr : MenuStr);
procedure  PrepMenu(var x1,x2,y1,y2 : Byte; var mstr : MenuStr;
                                  lne,num : Byte);
procedure  GetOption(x1,x2,y1,y2 : Byte; mstr : MenuStr; num : Byte;
                                 var submenu : CharSet; var ch : Char);
procedure  SearchSubMenu(rstr : MenuStr; menu :PtrToMenu; var lne : Byte);
procedure  GetAnsStr(x1,y1 : Byte; var rstr : MenuStr);
```

รูปที่ 5 แสดงส่วนหัวของโปรแกรมย่อยในยูนิต

แรกของรายการนั้น การดำเนินการดังกล่าวมานี้เป็นหน้าที่ของ GetOption โดย

- ตรวจหาอักษรนำตัวแรกของแต่ละรายการเก็บไว้ใน optset ซึ่งเป็นตัวแปรเซตของอักขระ และใน optChar ซึ่งเป็นตัวแปรสตริงเรียงจากรายการแรกไปยังรายการสุดท้าย
- หากรายการใดมีเครื่องหมายจุด จุดจะเก็บอักษรตัวใหญ่ของรายการนั้นไว้ใน submenu ซึ่งเป็นตัวแปรเซตของอักขระส่งกลับไป
- หากผู้ใช้กดคีย์อักษรจะทำให้เป็นอักษรใหญ่ แล้วตรวจว่าอยู่ใน Optset หรือไม่ ถ้าใช่จะส่งกลับไปใน ch เป็นการเลือกรายการนี้
- หากผู้ใช้กดคีย์ลูกศรลงหรือขึ้นจะเพิ่มหรือลดค่าใน y ซึ่งเมื่อเริ่มต้นจะชี้ที่รายการแรก ถัดจากนั้นก็เป็นรายการต่อไปหรือย้อนหลัง และเลื่อนเคอร์เซอร์ทึบไปยังรายการที่ต้องการ
- เมื่อผู้ใช้กดคีย์ Enter (เป็นการเลือกรายการนั้น) จะนำค่า y ไปเป็นอินเดกซ์ของสตริง optChar ได้ตัวอักษรใหญ่ของรายการนั้นกลับไปใน ch
- ท้ายสุดคือให้เก็บค่า y ไว้ใน oy ของ MenuDat

● **SearchSubmenu**

ค้นหาสตริงคือเมนูรองที่มีชื่อ (รายการแรก) ตรงกับค่าใน rstr (คือ retstr ใน PullDownMenu) เช่น FLF เป็นต้น ได้ค่าเป็นอินเดกซ์ของอะเรย์ของสตริงกลับไปใน lne

● **GetAnsStr**

เพื่ออ่านสตริงที่ผู้ใช้จะป้อน ตามที่กล่าวใน Prompter

ตัวแปรในยูนิต

ความสามารถในการสร้างเมนูซ้อนทับกันได้หลายชั้นนี้จำเป็นต้องมีตัวแปรสำหรับเก็บค่าต่าง ๆ ของเมนูไว้ (หรือไม่ก็เขียนเป็นโปรแกรมย่อยเรียกด้วยตนเอง) ปัญหาคือจะกำหนดไว้ในสกุลหรือนอกสกุล

ผมเห็นว่าหากกำหนดไว้ในสกุลจะเปลืองเนื้อที่หน่วยความจำมาก เพราะหน้าต่างแต่ละบานจะต้องมีตัวแปรนี้ ซึ่งที่จริงแล้วสามารถใช้ร่วมกันได้ เพราะการใช้หน้าต่างต้องเปิดใช้ทีละบาน การเรียกใช้เมนูก็จะใช้ทีละหนึ่งชุดเท่านั้น โอกาสที่ผู้เขียนโปรแกรมประกอบจะให้เปิดใช้หน้าต่างบานหนึ่งเพื่อเรียกใช้เมนูค้างไว้ แล้วไปเปิดหน้าต่างอีกบานหนึ่งเรียกใช้เมนูอื่นคงมีน้อยมาก (แต่ก็จะต้องเตรียมไว้)

จึงได้กำหนดตัวแปร MenuDat ไว้เป็นส่วนกลาง แต่นำไปไว้ในส่วน Implementation ดังรูปที่ 6 เพื่อกันไม่ให้ผู้เขียนโปรแกรมประกอบใช้ตัวแปรนี้ได้โดยตรง ตัวแปร MenuDat เป็นตัวแปรขนาด 20 ช่องของเรคอร์ด ประกอบด้วยฟิลด์ดังนี้คือ

-buf เป็นพอยน์เตอร์ไปยังหน่วยความจำที่จะเก็บภาพบนจอภาพเดิมตรงที่จะนำเมนูมาแสดง เพื่อนำมาคืนให้เมื่อเลิกใช้เมนูแล้ว เสร็จจากนั้นจะยกเลิกหน่วยความจำนี้

- numbyte คือจำนวนไบต์ของหน่วยความจำนี้ เพื่อใช้ในคำสั่ง GetMem กับ FreeMem

- xx1 ถึง yy2 คือค่าโคออร์ดิเนตของเมนูรวมกรอบ

- olne คืออินเดกซ์ไปยังเมนู (สตริง) เดิมก่อนที่จะนำเมนูใหม่มาซ้อนทับ (ถ้าเป็นเมนูแรกค่านี้ไม่ใช้)

- oy คือค่า y ซึ่งก็คือตำแหน่งของเคอร์เซอร์ทึบของเมนูเดิม (เมนูแรกไม่ใช้ค่านี้)

สำหรับจำนวนช่องของอะเรย์กำหนดไว้ถึง 20 ช่อง ซึ่งความจริงแล้วเมนูไม่ควรซ้อนทับกันมากไปกว่า 7-8 ชั้น ก็เพื่อหากผู้เขียนโปรแกรมประกอบจะให้เรียกใช้เมนูในหน้าต่างหลายบานพร้อมกันดังกล่าวแล้ว

นอกจากตัวแปรอะเรย์ MenuDat แล้วได้กำหนดตัวแปร midx เพื่อใช้เป็นอินเดกซ์ของอะเรย์นี้ โดยจะให้ค่าเริ่มต้นเป็น 1 (อยู่ใน begin end. ของยูนิต)

```
VAR     menuDat : ARRAY[1..20] OF RECORD
                    buf : RowPtrT;
                    numbyte  : Word;
                    xx1,xx2,yy1,yy2 : Byte;
                    olne,oy : Byte;
                              END;
        midx : Byte;
```

รูปที่ 6 ตัวแปรสำหรับเก็บข้อมูลของเมนู

```
UNIT  PRIME;
Interface

CONST
{ Control Key }
  null = #0;   bel = #7;   bs = #8;   lf = #10;   ff = #12;
  cr   = #13;  esc = #27;

  ctrlA = #1;    ctrlB = #2;    ctrlC = #3;    ctrlD = #4;
  ctrlE = #5;    ctrlF = #6;    ctrlG = #7;    ctrlH = #8;
  ctrlI = #9;    ctrlJ = #10;   ctrlK = #11;   ctrlL = #12;
  ctrlM = #13;   ctrlN = #14;   ctrlO = #15;   ctrlP = #16;
  ctrlQ = #17;   ctrlR = #18;   ctrlS = #19;   ctrlT = #20;
  ctrlU = #21;   ctrlV = #22;   ctrlW = #23;   ctrlX = #24;
  ctrlY = #25;   ctrlZ = #26;


{ Function Key }
  F1 = #59;  F2 = #60;  F3 = #61;  F4 = #62;  F5 = #63;
  F6 = #64;  F7 = #65;  F8 = #66;  F9 = #67;  F10 = #68;

  Home = #71;   Endd = #79;   PgUp = #73;   PgDn = #81;
  lArrow = #75;  rArrow = #77;  uArrow = #72;  dArrow = #80;
  ins = #82;  del = #83;

  ctrlHome = #119;  ctrlEndd = #117;  ctrlPgUp = #132;  ctrlPgDn = #118;
  ctrlLeft = #115;  ctrlRight = #116; ctrlPrtSc = #114;

  shifHome = #55;   shifEndd = #49;   shifPgUp = #57;   shifPgDn = #51;
  shifLeft = #52;   shifRight = #54;  shifUp   = #56;   shifDown = #50;

{ Normal Key }
  space = ' ';  comma = ',';  colon = ':';  semi = ';' ;  stop = '.' ;
  nullstr = '';

{ Screen attribute }
  norm = $07;  rev = $70;  inten = $0F;  blink = $87;  undlne = $01;


Implementation

END.
```

รูปที่ 7 ค่าที่กำหนดไว้ในยูนิต PRIME

เมื่อหน้าต่างใดเรียกใช้เมนูค่า midx จะเพิ่มเป็นหนึ่งทุก ๆ เมนูที่ซ้อนกัน และลดอีกหนึ่งทุก ๆ เมนูที่ยกเลิก

ยูนิต PRIME

ได้กำหนดค่าคงตัวไว้แทนรหัสควบคุมต่าง ๆ ทั้งในด้านชื่อรหัส (เฉพาะที่ใช้บ่อย) และคีย์ที่กดพร้อมด้วยค่าคีย์ของคีย์ฟังก์ชันต่าง ๆ ไว้ในยูนิต PRIME ดังรูปที่ 7 เพื่อใช้ในการเขียนโปรแกรมโดยทั่วไป และนำมาใช้ในโครงงานนี้ด้วย

ตัวอย่างโปรแกรมใช้งาน WN100H

พอมาถึงโปรแกรมใช้งานที่ผมอยากจะยกตัวอย่าง ผมรู้สึกอัดอัดใจพอสมควร เพราะโปรแกรมที่จะทำให้ดูสมเหตุสมผลนั้นจะยาวมาก จึงขอยกตัวอย่างเป็นโปรแกรมง่าย ๆ ก่อนครับ โดยสมมติว่าในโปรแกรมนี้มีโปรแกรมใช้งานอยู่ 2 โปรแกรมคือ SubProgramOne กับ SubProgramTwo นำมาเขียนรวมกันโดยมีท่อนแรกของโปรแกรมอยู่ในรูปที่ 3 และมีโปรแกรมหลักดังรูปที่ 8 รวมโปรแกรมทั้งสองนี้เข้าด้วยกัน ตามรูปนี้ท่อนแรกเป็นการเตรียมหน้าต่าง window1 ที่จะให้ SubProgramOne เป็นผู้ใช้ ขอให้สังเกตว่าได้ต่อ OptionCall เข้ากับ DoOption และ InfoWndWidHi เข้ากับ RecWndWidHi ซึ่งอยู่ในรูปที่ 9

ตามรูปที่ 9 ท่อนบนเป็นการ

```
begin  { MAIN }

  ClrScr;  Write('Start by press any key');  ch := ReadKey;

  window1.Init(41,79,5,20,'SUBPROGRAM ONE');
  window1.ConOptionCallTo(DoOption);
  window1.ConInfoWndWidHiTo(RecWndWidHi);
  window1.ShutWindow;

  window2.Init(2,35,2,24,'NOTE PAD');
  Window2.ShutWindow;

  REPEAT
    window1.ReOpenWindow;
    SubProgramOne;
    if  opt = 'E'  then  SubProgramTwo;
  UNTIL  opt = 'Q';

write('End of program press ENTER ');  readLn;
end.
```

รูปที่ 8 โปรแกรมหลักของโปรแกรมตัวอย่าง

```
{$F+}
procedure  DoOption(var rstr : MenuStr; var  ch : Char);
var   promp,finame,answer : MenuStr;
begin
  promp   :=  '==== Enter file''s name ====';
  finame  :=  'c:\pascal55\';
  answer  :=  'press ENTER';

  if  rstr = 'FLDO'  then
    begin  ch := Esc;
           window1.Prompter(promp,finame);
           WriteLn(finame)  end;
  if  rstr = 'FLFF'  then
    begin  ch := Esc;
           window1.Prompter(' Comment allez vous',answer) end;
end;

procedure  RecWndWidHi(wwid,whi : Byte);
begin
  WriteLn('This window is ',wwid,' width');
  WriteLn('            and ',whi,' height');
end;
{$F-}


PROCEDURE  SubProgramOne;
VAR  ch : Char;  astr : String;
BEGIN
  window1.ReOpenWindow;
  repeat
    opt := nullstr;
    ch := ReadKey;  if  ch = null  then  ch := ReadKey;
    if  ch = F10  then
      begin  Window1.PullDownMenu(5,5,@PrgMenu1,opt);
             WriteLn('Option = ',opt)  end
    else  ReadLn(astr);
  until  (opt = 'E') or (opt = 'Q');
  window1.ShutWindow;
END;
```

รูปที่ 9 โปรแกรมที่เป็นส่วนของ SubProgramOne

กำหนดโปรแกรมย่อยในแบบ far call สองโปรแกรม โปรแกรมแรกคือ DoOption ซึ่งต่อ (ให้ค่า) อยู่กับ CallOption ซึ่งใน PullDownMenu เมื่อผู้ใช้เลือกรายการใดที่ไม่มีเมนูรองก็จะเรียกใช้ CallOption เท่ากับเป็นการเรียกใช้ DoOption โดยตรง

วิธีการเช่นนี้ทำให้โปรแกรมที่เขียนขึ้นก่อนสามารถเรียกใช้โปรแกรมที่ยังไม่ได้เขียนได้ ใน DoOption ค่าที่รับมาคือ rstr เป็นสตริงเก็บค่าคีย์ที่ผู้ใช้กดในการเลือกรายการ เช่น ในกรณีแรกแสดงว่าผู้ใช้เลือกรายการ File, Load, Disk และ Other ตามลำดับ และในโปรแกรมนี้สมมติว่าต้องการชื่อไฟล์ จึงเรียกใช้ Prompter ให้ผู้ใช้ป้อนชื่อไฟล์ดังแสดงในรูปที่ 4 แต่ไม่ได้นำไปดำเนินการใด ๆ ต่อ

ส่วนค่าที่ส่งกลับคือ ch ได้ให้ค่าเป็น Esc กลับไป เพื่อยกเลิกรายการ Other ทำให้คงอยู่ที่เมนูของรายการ Disk ทำให้ผู้ใช้สามารถเลือกรายการ Pas, Exe, Com หรือแม้แต่ Others เองได้อีก

สำหรับโปรแกรมย่อย RecWndWidHi ซึ่งต่อ (ให้ค่า) อยู่กับ InfoWndWidHi จะทำงานทุกครั้งที่เปิดใช้หน้าต่างนี้ โดยรับค่าความกว้างและความสูงของหน้าต่างมาแต่ไม่ได้ดำเนินการใด เพียงแต่แสดงออกทางจอภาพเป็นการทดสอบเท่านั้น

ในโปรแกรมย่อย SubProgramOne ตามที่ถือว่าเป็นโปรแกรมใช้งานโปรแกรมหนึ่ง ผมเพียงแต่ให้อ่านคีย์มาตรวจว่าเป็นคีย์ฟังก์ชัน F10 หรือไม่ ถ้าใช่จะเรียกใช้ PullDownMenu โดยผ่านแอดเดรสของ Prgmenu1 (ตามรูปที่ 3) ไปให้ ถ้าผู้ใช้เลือกรายการที่มีการเรียกใช้ CallOption ก็จะมาทำงานที่ DoOption โดยอัตโนมัติ แต่ถ้า DoOption ไม่ดำเนินการใดก็จะส่งกลับมาในตัวแปรสตริง opt ซึ่งในโปรแกรมนี้ผมเพียงแต่ให้นำค่าออกแสดงเท่านั้น

อีกโปรแกรมหนึ่งคือ SubProgramTwo ตามรูปที่ 10 ให้ตรวจคีย์ 2 คีย์คือ F9 กับ F10 ทำให้มีเมนูได้ 2 เมนู และแสดงการผ่านค่าแอดเดรสของสตริงแทนที่จะเป็นแอดเดรสของอะเรย์ของสตริง

ขอย้อนกลับไปที่โปรแกรมหลักรูปที่ 7 หลังจากที่เตรียมหน้าต่างแล้ว โปรแกรมได้ให้เข้าทำงานในวงรอบ repeat เริ่มด้วยการเรียกใช้ SubProgramOne ทำงานดังที่กล่าวมา หากผู้ใช้เลือกรายการ Edit ก็จะเรียกใช้ SubProgramTwo และจะทำงานในวงรอบนี้จนกว่าผู้ใช้จะเลือกรายการ Quit ในโปรแกรมใดก็ได้ ทำให้จบโปรแกรม

ทั้งหมดนี้เป็นแต่เพียงตัวอย่างเท่านั้นครับ และถ้าจะให้เข้าใจดีคงต้องทดลองรันโปรแกรมนี้ครับ (มีบริการในซอฟต์เมล์) ในฉบับหน้าจะเป็นหน้าต่างแบบเคลื่อนย้ายได้ ย่อ-ขยายได้ เป็นหน้าต่างที่ซ้อนทับกันได้ และให้ผู้ใช้สามารถหมุนวนหน้าต่าง หรือเลือกหน้าต่างที่ต้องการขึ้นมาอยู่ด้านบนให้ครึกครื้นสมเป็นฉบับหน้าต่างครับ

```
CONST  Funct9 : MenuStr =
  'PERSON INFORMATION,Name,Address,Telephone number,Birth date,Office';
       Funct10 : MenuStr =
  'FILE,Load,Save,Rename,Change dir,Exit,Quit';


PROCEDURE  SubProgramTwo;
VAR   ch : Char;  astr : String;
BEGIN
  window2.ReOpenWindow;
  repeat
    opt := nullstr;
    ch := ReadKey;  if  ch = null  then  ch := ReadKey;
    if  ch = F9  then
      begin  Window2.PullDownMenu(2,3,@Funct9,opt);
             WriteLn('Option = ',opt)  end
    else if  ch = F10  then
      begin  Window2.PullDownMenu(8,2,@Funct10,opt);
             WriteLn('Option = ',opt)  end
    else  ReadLn(astr);
  until  (opt = 'E') or (opt = 'Q');
  window1.ShutWindow;
END;
```

รูปที่ 10 โปรแกรมที่เป็นส่วนของ SubProgramTwo